Truck Platooning: week 1

Changyao Zhou, Jiaxin Pan

December 13, 2021

1 Introduction of the project

The goal of this project is to chase a chosen target vehicle by estimating the steering angle and throttle input of the ego-vehicle, given the velocity of both vehicles.

The control values (steering angle and throttle) computed by Model Predictive Control (MPC) are used as ground-truth. MPC is an advanced method of process control that is used to control a process while satisfying a set of constraints[WIK]. It takes a prediction horizon instead of a single time step into account, and aims to get an optimal control result minimizing the objective function within the prediction horizon. For learning the MPC controller, we take [MPC] as reference.

The project is designed to be realized in the following steps:

1). Control the ego-vehicle with Model Predictive Control (MPC) in the Carla simulator

2). Given object velocity, ego-velocity and RGB image, train a neural network to control the ego-vehicle by using the MPC output as ground-truth

3). Further improvement by changing the input of the neural network

2 Overview of this week

To train the neural network, first the ground-truth data is required. In our project, the neural network should control the ego-vehicle as model predictive control.

Therefore, in this week, we tried to get familiar with the principle of model predictive control. We implemented the vehicle-chasing task in a simple model and tried to convert the outcome to the Carla simulator.

All the videos are stored under `https://syncandshare.lrz.de/getlink/fiXpJvvzfbfg2Y3ZTkGkJzHZ/week1`.

3 Vehicle-chasing with MPC in a Simple Model

3.1 Overview

In our task, given the plant model and the cost function, with the trajectory of the target-vehicle already known, the MPC controller can predict the ideal trajectory of the ego-vehicle by following the target vehicle with some distance. The minimum/maximum throttle and steering angle are set as constraints.

3.2 Design the Plant Model

In the plant model, the next state is predicted based on the current state. The predicted next state is used to calculate the cost in the cost function. Since the goal of first simple model is to test if the vehicle chasing task is achievable by MPC, the plant model is designed as simple as possible. The plant model is designed as following:

$$x_{t+1} = x_t + v_t * dt * cos(psi_t) \tag{1}$$
$$y_{t+1} = y_t + v_t * dt * sin(psi_t) \tag{2}$$
$$v_{t+1} = v_t + a_t * dt - v_t/25.0 \tag{3}$$
$$psi_{t+1} = psi_t + v_t * dt * tan(beta)/length \tag{4}$$

Where x_t and y_t represent the current x and y location of the ego-vehicle, v_t is the current velocity and psi_t is the current yaw angle. The control value are a_t and $beta$, which are the input throttle and the input steering angle respectively.

3.3 Design the Cost Function

In the cost function, both the position and angle costs are taken into account. Since both the target-vehicle and ego-vehicle are moving, their states are changing in every step. Therefore, the accumulated cost is calculated as a sum of cost from each step within the prediction horizon, depending on the states of target-vehicle and ego-vehicle in that step.

To follow the target car, there must be a proper distance between the target-and ego-vehicle to avoid the collision especially there is a sudden stop with the target car. In our cost function, the fixed distance is set to be the same as the length of the vehicle. In the next step, we might use the desired car distance which is related to the velocity of target-vehicle.

The position cost is calculated separately for x-and y-axis with absolute distance:

$$cost_x = (abs((x_{ref} - distance_x) - x_{ego})) \tag{5}$$
$$cost_y = (abs((y_{ref} - distance_y) - y_{ego})) \tag{6}$$

Where $distance_x$ and $distance_y$ are the projections of distance on the x-axis and y-axis, respectively. They can not be easily calculated either by using the yaw angle of the target-vehicle or by using the yaw angle of the ego-vehicle. Since they lead to either a larger turning radius or a smaller turning radius by turning. Therefore, in our cost function, $distance_x$ and $distance_y$ are calculated with weighted average of the output distances for this two angles:

$$distance_x = distance * (cos(psi_{ego}) * 0.3 + cos(psi_{target}) * 0.8) \tag{7}$$
$$distance_y = distance * (sin(psi_{ego}) * 0.3 + sin(psi_{target}) * 0.8) \tag{8}$$

The angle cost is calculated based on the difference between the yaw angle of the target-vehicle and the ego-vehicle:

$$cost_{angle} = (psi_{ego} - psi_{target})^2 \tag{9}$$

3.4 Output Analysis

We test our model by following a target-vehicle which traverses several typical trajectories: go straight, turn left, turn right, accelerate, decelerate, sudden stop.

The result can be seen in video 'simple_model_test.mp4'. The ego-vehicle first chase the target-vehicle and then follow it as desired. The ego-vehicle can follow the chase and follow the target-vehicle perfectly even if there is a sudden stop or a sharp turn. The distance is proved to be reasonable to avoid collision even if the target-vehicle stops suddenly.

4 Vehicle-chasing with MPC with Carla simulator

The procedure is supposed to be similar to predict the inputs of the ego-vehicle with MPC. We first tried to implement MPC in a online way, which means compute the control values while following the target-vehicle in autopilot mode. However, with the same model we tested above, the result seems unreasonable. The ego-vehicle keeps shaking violently. However, there is a problem that the plant model in the Carla simulator seems to be very different from our simple model, especially for calculating the velocity with throttle value. The MPC output of the throttle is always 1.0, which means the acceleration is very high all the time, but the ego-vehicle still can not even chase the target-vehicle. However, the recorded throttle value of the target-value is much smaller. We suppose the possible reason might be that during the online control, the MPC have no information about the target-vehicle in later time steps, thus it can't do 'prediction control'.

Therefore, we decided to compute the MPC output in an off-line way, which means first run the target-vehicle with 'autopilot' and record the steering and throttle values. Then calculate the control value of the ego-vehicle with MPC for all time-steps. After that, both the recorded control values for the target-vehicle and the predicted control values for the ego-vehicle should be used as input value to control the two vehicles and check if everything works well.

The test video can be seen with 'Carla_offline.mp4'. We used the plant model from our simple model. The output seems to be quite reasonable, however, the constraints on the throttle is set to be [-10, 10] here, while in Carla simulator it should be [0,1]. So we have the problem about how to integrate the result of our simple model into the Carla simulator.

5 Conclusion and future work

In this week, we first tried to apply MPC controller to our vehicle-chasing scenario in a simple model. The MPC controller seems to be capable for this task since the output is quite desirable. In the next step, we should apply the MPC controller in the Carla simulator. What we currently are lack for is the specific plant model in the Carla simulator, especially the relation between throttle, brake, and the velocity.

References

[MPC] Model Predictive Control. https://github.com/WuStangDan/mpc-course-assignments.

[WIK] Model Predictive Control. https://en.wikipedia.org/wiki/Model_predictive_control.